AD-787 684

A METHODOLOGY FOR SELECTING AND REFINING MAN-COMPUTER LANGUAGES TO IMPROVE USERS' PERFORMANCE

John F. Heafner

University of Southern California

Prepared for:

Advanced Research Projects Agency

September 1974

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ISI/RR-74-21	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD 787684
4. TITLE (and Subtitle) A METHODOLOGY FOR SELECTING AND REFINING MAN-COMPUTER LANGUAGES TO IMPROVE USERS' PERFORMANCE		5. TYPE OF REPORT & PERIOD COVERED Research Progress report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John F. Heafner		8. CONTRACT OR GRANT NUMBER(s) DAHC 15 72 C 0308
9. PERFORMING ORGANIZATION NAME AND ADDRESS USC Information Sciences Institute 4676 Admiralty Way Marina Del Rey, California 90291		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order #2223
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, Virginia 22209		12. REPORT DATE September 1974
		13. NUMBER OF PAGES 64
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) ----------------------
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE ----------------------

16. DISTRIBUTION STATEMENT (of this Report)

This document approved for public release and sale; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

----

18. SUPPLEMENTARY NOTES

Thesis proposal

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

language extension methods for problem-oriented languages, man-machine performance modeling, modeling, modeling computer users and services, monitoring, performance measurement, statistical analysis of man-machine communication, statistical users' performance evaluation.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(OVER)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This report describes a methodology (supported by a software package) to model, measure, analyze, and evaluate users' performance in a message communication system environment. The theses of the report are: 1. that models of users and services can be accurately used as predictors in selecting a language form, for an application, which will result in high users' performance, and 2. that such a language form is only an approximation (in terms of yielding optimal user's performance) due to within variances of user and service-classes, hence individual, on-line regulation of language constructs is necessary to further improve performance.

This report develops appropriate models and algorithms, and states hypotheses relating the interactive effects of users, services, language forms, and other variables important in man-machine discourse. An experiment design is presented, which tests the major hypotheses.

ARPA ORDER NO. 2223

ISI/RR-74-21
September 1974

John F. Heafner

A Methodology for Selecting and Refining Man-Computer Languages to Improve Users' Performance

INFORMATION SCIENCES INSTITUTE

UNIVERSITY OF SOUTHERN CALIFORNIA

4676 Admiralty Way/Marina del Rey/California 90291
(213) 822-1511

THIS RESEARCH IS SUPPORTED BY THE ADVANCED RESEARCH PROJECTS AGENCY UNDER CONTRACT NO. DAHC15 72 C 0308, ARPA ORDER NO 2223 PROGRAM CODE NO 3030 AND 3P10

PREFACE

The Information Automation project at USC/Information Sciences Institute is currently developing methods to automate various information handling tasks, with particular emphasis on message processing for military command, control, and communications. The project, sponsored by ARPA, is an integral part of the client's and ISI's overall program to explore the use of computer technology and methodology in military environments.

This report is one of a planned collection of reports that describes the current status and future plans of the Information Automation project. Specifically, this report describes the project element called the User Monitor, whose purpose is to investigate the theory of man-machine dialogue.

The project has designed a three-part instrumentation and adaptation study to examine users' performance as related to 1) man-machine language forms, 2) algorithms used to execute service functions, and 3) resource allocation policies within the operating system. The first of these parts, concerned with the theory of man-machine dialogue, is termed the User Monitor. This report defines the purpose of the User Monitor and presents a rudimentary specification intended to hold throughout initial experimentation. ***Some parts of the User Monitor are specified in enough detail to be suitable for implementation; other parts, not as well thought out, are only sketched.***

Those who wish only an overview of the User Monitor and its relations to the project's purposes should read the Introduction and Overview, as well as Chapter 3, which describes the hypotheses of the study. For those not familiar with the basic techniques of statistical analysis, Chapter 12 briefly defines those techniques suggested for use.

A cursory definition of other project elements is given to make it possible to read and understand this report as a separate entity. For a more comprehensive discussion of other project elements, the reader is referred to project documentation noted in the bibliography.

SUMMARY

Briefly, this report describes a method (and a software package to support it) to model, measure, analyze, and evaluate users' performance in a message communication system environment. Its theses are, first, that models of users and services can be accurately used as predictors in selecting a language form for an application that will result in high users' performance, and, second, that because such a language form is only an approximation (in terms of yielding optimal users' performance) due to "within" variances of user- and service-classes, individual on-line regulation of language constructs is necessary to further Improve performance. The study develops appropriate models and algorithms and states hypotheses relating the interactive effects of users, services, language forms, and other variables important in man-machine discourse. An experiment design is presented which tests the major hypotheses.

Chapter 1 presents the context of the investigation of dialogue theory, which deals with the consolidation and automation of military message processing. For some time, the computer science community has recognized the need for what has been labeled a "transfer mechanism," i.e., the means by which thoroughly researched tools can be infused into nonresearch environments. More recently, we have begun to realize that this transfer mechanism is multidimensional. This study concerns itself with one facet of the mechanism, namely, how to insure that the tool (in this case, a military message processing service), once installed, is used effectively. More precisely, we ask, "How do we make the service effective for a large number of individual users who have different skills, abilities, educational backgrounds, and so forth?" It follows that a single man-computer language form does not suffice for optimum performance for all such users. Hence we wish to improve users' performance in two ways: first, a tractable language form is chosen, based on prior modeling of users' traits and the message service's idiosyncrasies; second, the language selected is tailored to each user by refining it on-line with the help of the user himself and the results of a model of his performance. Both initial selection and later refinement are based on classical techniques of statistical analysis. Chapter 1 concludes with a concise definition of each of the major components of the prototype message service.

Chapter 2 compresses the study into a statement of its major goals, such as developing the means to find best languages. These give rise to ancillary goals such as why, in terms of the formal properties of the user-service transactions, the particular languages chosen are best. Such goals imply the need for models, i.e., of users to be served, of services to be offered, and of users' performance.

Chapter 3 formalizes the goals as explicit hypotheses concerning the effects on performance of language forms, prior experience, training methods, and so forth. A number of hypotheses are suggested in order to illuminate the breadth and depth of the questions under study. However, only the most interesting of them will be tested in initial experimentation.

Chapter 4 then explains why all hypotheses are not to be tested at once. It chooses those that are to be tested, describes a factorial design for the experiment, suggests the method of training, and describes the task environment control.

Summary

With the goals of Chapter 2 formulated into hypotheses in Chapter 3, Chapters 5 and 6 develop models identified in the Overview. We wish to apply the user model as a predictor of language forms and other variables. The model presented is a psychological model of users' traits, which are of interest because of the users' information processing behavior and ability to determine man-computer performance. The model is applied by administering (off-line) to the potential user a battery of psychological tests, with the expectation that the results will relate performance to variables such as language forms. The psychological model is defined, and testing and scoring are specified. One test, for example, deals with manifest anxiety, a measure used to indicate the amount of feedback necessary in the man-machine dialogue. Two other models, i.e., functional and demographic, are mentioned but not explicated in detail.

Chapter 6 defines the service model to be used, which essentially classifies and codifies each type of request and response in terms of some attributes reflecting the user performing his task. Since only one service is to be offered, the service model does not play a direct part in our experiment. Still, a quantification scheme will be developed for future application of the model, since the service model is an integral component of the methodology.

Clearly, this study must involve several language forms in order to test their effects. Chapter 7 names those forms, provides a notation useful in describing observations of man-machine transactions, gives a representation of the transactions using the notation, and defines some terms used in referring to the performance modeling.

Chapter 8 defines performance indicators, dialogue properties, and system properties, and provides a notation for their reference. Metrics of performance must measure production, interaction rate, and errors as well as subjective quality. The chapter delineates activity measures and their statistics, used as performance indicators. Evidently, we wish to know why, in terms of the actual interactions, the models work. Hence, we define the format and complexity of the dialogue as a set of items, along with the activity measures observed, when sampling user service interactions. Also identified are system properties (such as the time of day and the load average) that are either to be controlled during experimentation or to have their effects removed during analysis.

Chapter 9 continues the discussion of on-line language modification and extension, the rationale for which was given in Chapters 1 and 2. Three cases arise for which language changes are suggested to the user. The first, called ineffective dialogue, is that in which the user cannot proceed because he is not entering a command correctly. The persistence of such activity causes the system to try to identify the spurious command, determine why it causes trouble, and attempt remedial action. The second, called inefficient dialogue, stems from transactions that do not prohibit useful work but do lead to poor performance. The system detects these elements and makes the necessary adjustments. The third, called recurrent dialogue sequences, involves detecting a user's habitual actions, then taking the initiative to reduce the amount of protocol necessary to accomplish the task (e.g., if the user frequently logs on and then reads his mail in the morning, then the ten or so operations involved can be relaced by a single new operation). The system also compiles significant recurrent dialogue sequences for several individuals in a class such that their habits may be useful to another user. The chapter presents the various algorithms used in each of the three cases.

Summary

Chapter 10 outlines the on-line analysis and the post-analysis.

The user profile is a data base consisting of separately identifiable files, which is a concise record of users' behavior and experience with the message service. Data in the user profile are used for much of the training and analysis. Chapter 11 describes each file in the profile, together with its purpose, format, and construction procedure. Other important data bases are similarly described.

Chapter 12 is included as a complete correspondence to Chapter 3. It briefly describes the statistical analysis techniques necessary to test each of the hypotheses, then gives a very brief description of each hypothetical experiment in terms of data organization and necessary statistical methods.

The Appendix, a dictionary of terms used throughout the report, might well be scanned before the body of the report is read.

ACKNOWLEDGMENTS

The author would like to thank the following persons for their careful review and helpful comments on this report: Dr. Gabriel Groner of The Rand Corporation, Santa Monica, California, and Drs. Walter Ryder and Stephen Kimbleton of USC/Information Sciences Institute.

The author would also like to especially thank Dr. James Carlisle of USC/Information Sciences Institute for his penetrating insight into the organization of the statistical analysis problems implied by the User Monitor and his help in crystallizing the author's thinking on that organization. Thanks are also owed to Dr. Gerhard Hanneman of the USC Annenberg School of Communications for his help and guidance in selecting psychological tests for the User Model.

CONTENTS

1. INTRODUCTION

The Information Automation project [1] is currently developing methods to automate various information handling tasks with particular emphasis on message processing for military command, control, and communications [2]. One of the prominent military requirements for such automation is that any given system is to be used by a very large number of people with diverse skills, abilities, education, and motivation. It follows that one would not expect each individual user to perform at his best if the entire population used a single man-computer language style. There does not exist today an effective procedure for determining and then continuously altering a language to yield highest performance for each particular user.

From the viewpoint of both economics (costs) and the user's gratification (benefits), it is desirable to maximize the user's performance. Hence this report concerns the development of a methodology for selecting and for dynamically altering man-machine dialogue forms to optimize a given user's achievements. The user's time is assumed to be very important. From this it follows that the reason for studying the effects of language forms on a user's performance is that we anticipate a significant payoff from the results.

Let us consider selection and alteration separately. How does one select the best language form for a particular user in some specified environment? This study intends to provide a way to choose the language by first modeling the user and the service, then--using the results of these models--predicting the appropriate communication style. But the communication form chosen is only an approximation of the best one for any user in a given class, and its closeness of fit is determined by the accuracy and resolution of the modeling and by the variation within the groups to which the models are applied. Once a language form has been selected, to further refine particular dialogue elements the communication between user and service should first be observed and analyzed. The system can then offer language alterations which, based on the analyses, are presumed to result in better user performance.

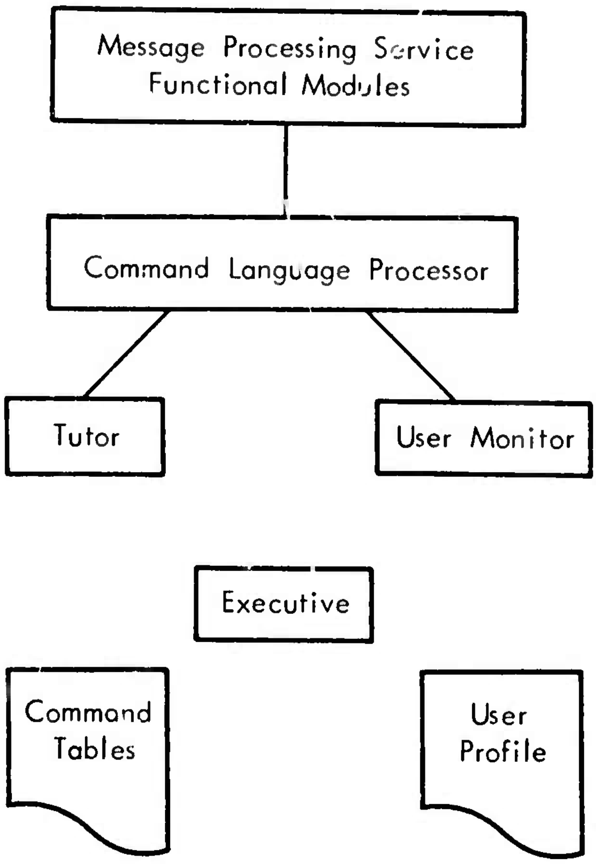
In particular, the plan is to proceed as follows. Languages of the forms supported will be pre-tested to ensure that they are equally representative for use by the real target population (military users). A first experiment will be designed and conducted using college student subjects. The kind of service, task environment, on-line dialogue regulation, training methods, helping aids, and levels of user experience will be held constant; the experiment will vary types of users and language forms. Results of the analysis will confirm or reject specific hypotheses about relationships among these variables as modeled and measured. Evaluation of the method and recommendations for subsequent experiments involving military personnel will then follow.

Note that the emphasis here is on the methodology and that the use of student subjects in lieu of military personnel has several ramifications. The student population was chosen for its availability. Because the task definitions should somewhat reflect these subjects, one should not make strong inferences concerning military personnel based on experimentation with students. Follow-on experiments should be directed toward the real target population.

1. Introduction

To ensure that this report can be understood without necessarily reading the companion reports listed in the bibliography, the major program modules which make up the message processing environment are defined here. The five major components are shown, along with two important data bases, in Figure 1. The most obvious component is the message processing service Functional Modules. Their function is to execute the user's requests to compose, edit, send and receive messages, and so forth. Another module is the Command Language Processor (CLP), which parses user's input and, in general, mediates between the user and the message service. The Tutor provides on-line training for the user, and also--assisted by the User Monitor--helps the user tailor commands to his individual preferences. The User Monitor checks the user's performance and recommends (via the Tutor) alternate forms of dialogue to improve performance. It also conducts post-analysis to develop language selection methods that are described within this report. The Executive interfaces the other components to the operating system. It is responsible for various tasks such as I/O handling, process control, and checkpoint and restart. The command tables data base contains language-dependent commands for each supported language for each message service function. The User Profile is a collection of separately identifiable data bases that describes, for example, users' performance, training, and service usage.

1. Introduction



Message Processing Service Functional Modules
Command Language Processor
Tutor
User Monitor
Executive
Command Tables
User Profile

Figure 1. Components of the message processing environment

2. *OVERVIEW: GOALS AND MODELS*

STUDY OBJECTIVES AND THE MODELS THEY SUGGEST

How is a theory relating dialogue forms to types of users and kinds of services to be used? One wishes to provide meaningful answers to questions such as the following. A planned library service will directly serve clerical personnel and occasionally be used by the management staff. Which dialogue forms are best fitted for this environment? Within the set of language forms acceptable for this purpose, which can most naturally be extended by the user? How should the user be taught to use the service?

To answer such questions, the dialogue theory must embody proper models of the environment in question, consisting of (1) users to be served, (2) services to be provided, and (3) users' performance.

Thus, models of a user and a service may later be used as predictors for maximizing such criteria as language forms and training methods.

SOME PRIMARY GOALS OF THE STUDY

With these kinds of models in mind, one can restate the questions above as goals of an investigation of the theory of dialogue. Following is a general discussion of goals. The next chapter states explicitly the hypotheses with which this study is concerned.

One goal is to determine language forms yielding best performance given a user and service environment. An ancillary goal is to determine why, with respect to the formal properties of dialogue, these forms are best. The first goal suggests that one derive empirical laws relating language forms to types of users and kinds of services, such that they will allow one to choose the proper language constructs given the latter two. Consider experiments with fixed user types and service kinds, using different dialogue forms and instrumenting the resulting performances. With appropriate models and analysis, one can in the future select the best dialogue form for a communications environment.

Since it is presumably desirable to fine-tune (on-line) any chosen language, another goal is to determine language forms (yielding best performance) for naturally extending the user's knowledge. An ancillary goal here is to determine the actual working primitives or primary language elements of a given language.

Man's ability to express himself is limited in one sense by the richness of his vocabulary; thus one would expect his performance to relate to his knowledge of the language he is using. Indeed, one wishes to determine which language forms are most conducive to extension by the user. His knowledge level and its rate of growth can be determined by noting the language elements with which he is familiar. By random sampling, the frequency of use of these language elements (i.e., his real working vocabulary) can be calculated.

One determines how well the user can synthesize language constructs by looking at extensions he has defined, which portray his operational coalescence of service functions. In general, from his extensions, one can deduce qualities of language style (e.g., perspicuity, consciseness, etc.) with which he is most comfortable.

Furthermore, the composition of composite operations (i.e., replacing a sequence of user operations by a single operation that accomplishes the same functions) leads one to discover the appropriate level of primitives for a given user/service environment. That is, if one discovers among users statistically significant concurrence of compound operations, then one is led to conclude that these function sets should be primitive.

Yet another goal is to determine the best methods for user education and the most appropriate on-line aids: hardcopy manual and Tutor alone, these with user-initiated extensions allowed, or these with system-suggested extensions permitted. An ancillary goal is to determine if dynamic regulation of user service dialogue has a positive effect on performance.

Evidently, with appropriately selected experiments, one can determine a user's performance in relation to his knowledge of the language. Furthermore, one can determine his performance in relation to the way in which his knowledge was acquired. The tests suggested in this report permit us to answer these questions.

Stemming from the notion of tailoring language elements to individual users, another goal is to detect inferior performance and take remedial action in the form of altering the dialogue constructs. Three cases arise. The first case (mentioned above) is to recognize dialogue sequences that occur with significant regularity (recurrent dialogue sequences). Once identified, a composite form can be recommended to the user. The second case arises in which the user repeatedly mistypes a command. The User Monitor attempts to identify the spurious command and to suggest (via the Tutor) some alternate form. The third case involves the isolation of those language constructs that do not prohibit useful work but that lead to poor performance. Again, alternate forms are recommended through the Tutor.

Exploring the above goals quite expectedly suggests other more concrete hypotheses. The next chapter states these hypotheses that this program element shall attempt to validate.

3. *HYPOTHESES*

PURPOSE AND IMPLICATIONS

The purpose of this study is twofold. The first goal is to develop methods for predicting the most appropriate language forms and training methods for classes of users over a range of applications (the methods are to be substantiated by experiment). Yet any language form chosen for a uniform group of users is only a good approximation of that needed to ensure maximum performance by each user. Thus, the second goal is to develop a software mechanism to further refine the selected language, that is, refine it on-line in conjunction with the user.

This statement of purpose assumes a certain collection of "facts." These suppositions can be explicated as hypotheses, along with analysis procedures to validate them. These hypotheses and the analysis procedures provide the essential support of the assumptions underlying the study. The hypotheses are stated below. (Their rationale was given in the previous chapters; the analysis procedures are given in Chapter 12.)

In the following hypotheses, assume that variables not specifically under test are either fixed or that their effects are eliminated in the analysis (for particulars of the tests, see Chapter 4). For example, if one is interested in language form differences, the following is a typical experiment setting. Assume there are "m" test groups, "n" users in each group, with each group using a different language form. The following parameters remain constant: task description, environmental control, user type, service kind, and training method. (The effect of system variables such as load average is eliminated in the analysis.) An on-line experiment is conducted. Raw data measuring performance are collected on each user during the test. Statistics are computed from the data. A set of "k" performance indicators is determined from the statistics for each user. These performance indicators are the criteria variables in the analysis.

Our interests lie mainly in determining relationships between language forms and other variables. There are many interesting variables other than language forms for which hypotheses similar to those below can be stated simply by replacing "language forms" by the variable of interest. Examples of such variables are:

- o Training methods and on-line helping aids
- o Previous user's experience

Using these two variables in the hypotheses below would triple the analysis; thus they will not be explored in the initial experiment of this study. However, they are pointed out as possible follow-on experiments.

The first five hypotheses below will be tested in the initial experiment. Others--namely sets 6 and 7--are stated here merely because they are interesting hypotheses for later analysis. Some hypotheses in sets 6 and 7 are almost "free," since the data for their analysis will have been collected in the initial experiment.

HYPOTHESES

Hypothesis 1: User and Service Models

1. Users and services can be represented according to the models defined in this report, and the results can be used as predictors in selecting language forms.

The reader will note in Chapters 4 and 12 that user types will vary in this experiment, but service kind (military message processing) will not.

Hypothesis 1 constitutes the major thesis of this study. Ultimately, we want to use the methodology to this end. But, in addition to offering empirical evidence in support of the hypothesis, one would like to explain why the models work. Consequently, a number of subordinate hypotheses follow. Understandably, the "why" is a complex question; the following hypotheses seek a "shotgun" approach that, at best, only goes a little way toward providing satisfying answers.

Hypothesis 2: Overall Performance

2. Different language forms result in significantly different levels of user performance.

Hypothesis 3: Maximum Discriminants

For practical use, the User Monitor should collect data on a ***small*** number of performance indicators. Our initial experiment will make use of all the performance indicators defined in Chapter 8; post-analysis will then determine the most useful ones for similar future experiments.

3. There is a kernel set of performance indicators that adequately discriminates performance with respect to language form.

Hypothesis 4: Formal Dialogue Properties

Hypothesis 2 leads naturally to the next level of detail. Why, in terms of the attributes of user-service interaction, is one language form better than another?

4. The differences in users' performance (by varying language forms) can, in part, be explained by differences in the values of the formal properties (format and complexity) of the dialogue.

Hypothesis 5: Relationship of Language Forms and Training

There are meaningful interrelationships between language forms and training methods and helping aids. Of interest is the following hypothesis.

5. The learning and training time necessary to reach a plateau of performance differs significantly across language forms.

Hypothesis Set 6: Language Extensions

6A. Some language forms are more suitable than others for extension, that is, more macro operations are defined and employed by the user.

6B. Some are extended more uniformly, that is, users employ the same macros in a given language form.

6C. Multilingual users (i.e., those who have used other language forms with the message service) have a higher propensity for language extensions than those using only one language form.

6D. The number and kinds of primary language elements (i.e., those primitives and compounds that account for most of the man-machine interaction) vary according to language forms.

6E. Qualities of language style can be inferred from the dialogue properties of the primary language elements.

Hypothesis 6E is included here as an anecdotal hypothesis. One would hope that, in defining new languages, one could recognize distinct modes of expression which correlate well with performance indicators ascertained in hypothesis 3. This *ad hoc* analysis should be recognized as an interesting "fishing expedition," with the understanding that during the experiment probably not enough data will be taken to warrant a strong conclusion. Also, since styles are not known *a priori*, there is no way to rate or weight criteria.

6F. The number of primitives making up a compound dialogue element is related to its frequency of use.

6G. The length of time that a compound has been defined is related to its frequency of use.

Hypothesis 6F is mostly an interesting pedagogic device in the following sense. One would expect the frequency of use to be a function of the length of time that the (compound) transaction type had been defined. Hence, normalizing frequency on this basis might yield a linear relationship, whereas a plot of the data before transformation might be a quadratic or higher order function.

Hypothesis Set 7: Effectiveness, Efficiency, and Recurrency

Some portion of the user's commands are unrecognized by the CLP, but are identified by the User Monitor. Among those identified, some are then remedied by the Tutor and user.

7A. These proportions differ with respect to language form.

Similarly, some percentage of inefficient dialogue elements (i.e., those leading to poor performance but not prohibiting useful work) for which alternates are suggested are actually changed.

7B. This percentage, along with the intensity of suggestion required, differs according to language forms.

Again, some part of the recurrent dialogue sequences for which composite commands are suggested are actually accepted.

7C. The percentage accepted is also a function of language forms.

The analysis procedures for these hypotheses are sketched in Chapter 12. Chapter 4 defines the initial test design, task environment, and so forth, to test the first five hypotheses.

4. INITIAL EXPERIMENT

LANGUAGE PRETESTS

Developing methods for choosing language forms best suited to a particular user and communication environment necessarily requires that we conduct experiments using several kinds of languages. This chapter suggests how the experimental languages are developed.

One requirement is that we assure that the specific languages are indeed equally adapted to the way in which military users conduct their tasks. This is accomplished by defining "straw-man" languages and pretesting them by "protocol analysis." The results of that analysis are an important ingredient in the design of the actual languages used for experimentation.

What is the protocol analysis involved in language pretesting? Potential users (See "Action Officers" in [2]), such as CINCPAC* or perhaps AMC** personnel, are given the straw-man languages, and (with an analyst, but no computer system) go through a scenario as if they were performing a typical daily task using the language. They are invited--in fact instructed--to comment on the strengths and weaknesses of the service functions themselves, the basic syntax of the language, and particular idiosyncrasies such as parameter arrangements, abbreviations, etc., of individual operations. Then the language designer specifies (from these results and the straw-man languages) the actual languages to be used in Initial experimentation.

SCOPE OF THE FIRST EXPERIMENT

The goal of the experiment is to determine which variables are relevant to the models and what functional relationships exist between criterion and predictor variables. Our "workhorse" is the analysis of variance described in Chapter 12. It is most appropriate to the analysis of the effect of different variables and their interaction i.e., the effect of one in the presence of another.

Our analysis is to be multivariate; we are interested in the effect of many variables. A natural way to design the experiment would thus be to take an equal number of observations of every combination of variables. Summing the number of hypotheses and their attendant variables shows clearly that we would need a large number of conditions. This kind of multivariate experiment is called ***factorial design***; analyzing the variance of its results allows us to study the variables and their interactions.

The ***Latin square*** is a prevalent design for multiple variables commonly used to reduce the number of observations required by the factorial design. However, it makes a very important assumption that we are unwilling to concede, i.e., that the effects of interaction among variables are not significant. Some of our

----

*Commander In Chief, Pacific
Camp Smith Telecommunications Center
Oahu, Hawaii

**Department of the Army
Headquarters, Army Materiel Command
5001 Eisenhower Avenue
Alexandria, Virginia 22304

hypotheses, on the contrary, state that interactive effects *are* significant.

To resolve this dilemma--either charting an overly complex initial experiment or ignoring the possibly meaningful effect of variable "crosstalk"--we shall compromise by choosing the factorial approach and testing the first five hypotheses until we gain a deeper understanding of some of the fundamental interactions among variables, if any. Hence the first experiment will test a subset of the hypotheses; subsequent tests (perhaps using a Latin square design) might address additional hypotheses.

Thus it is desirable to combine as many tests as possible into a single, simple first experiment. Assuming techniques of analysis of variance, consider the theoretical model

$$M_i = U + T_{lf} + T_{ut} + T_{te} + f(T_{lf}, T_{uf}, T_{te}) + \epsilon.$$

where M_i represents the performance indicators delineated in Chapter 8, including the quality performance indicator, i.e., the amount of work correctly accomplished during testing. Further,

U = mean of observations of a performance indicator over all samples
T_{lf} = the effect of language form differences
T_{ut} = the effect of user types
T_{te} = the effect of training and experience
$f(T_{lf}, T_{uf}, T_{te})$ = the interactive effect of each of the cross-product terms
ϵ = the error term and residual of untested factors

The equation does not account for system properties such as time of day and load average, since these effects are eliminated (by partial regression) from the raw data. Hence, hypothesis 1 submits that $T_{lf} \times T_{ut}$ is significant, hypothesis 2 states that T_{lf} is significant, and hypothesis 5 hypothesizes a significant effect from T_{te}.

In addition to these three hypotheses, the design below allows us to test hypothesis 4 using either canonical correlation or factor analysis followed by multiple partial regression. Also, hypothesis 3 can be tested from this design by discriminant analysis. (These statistical analysis techniques are briefly described in Chapter 12.)

THE FACTORIAL DESIGN

To more readily discover the actual significant variances, one wishes to have a high ratio of cell size to the number of cells. The number of cells required to test the above hypotheses is four (see Fig. 2). Eight or ten subjects are to be included in each cell; this is no doubt a bare minimum cell size. Ideally, one would like to use, say, 20 or 30 subjects per cell.

	L1	L2
u_1	$\approx$ 8 subjects	$\approx$ 8 subjects
u_2	$\approx$ 8 subjects	$\approx$ 8 subjects

Legend:
- L1 = language form 1, e.g., English-English
- L2 = language form 2, e.g., Canonical Form-Menu
- u_1, u_2 = user types determined after psychological testing.

Figure 2. Factorial design experiment

TRAINING METHODS

An explicit specification of training methods is necessary to achieve independence of, and impartiality toward, language forms. We propose a split training session. The first part is given to all subjects (representing all groups). It stresses what is to be measured and thus what defines performance; it is independent of language form. A second language-specific part is given individually to each group with examples from the appropriate language. This part should be impartial toward languages in that examples of use do not favor the particular language being taught.

TASK ENVIRONMENT

Each college student subject will participate in six sessions of 1.5 hours duration per session, making a total of nine console hours per subject over a period of about two weeks.

Each subject will perform tasks from the same pool of message processing tasks, where the pool for each session is the same for all users. That is, there will be no session-to-session variation (other than random variation) in the nature of the tasks. Enough task scenario material (message composition, coordination, and sending and receiving messages) will be available for "daily work" so that no subject will finish the entire pool within the 1.5 hours. These tasks have not yet been defined. Subjects will be requested not to discuss among themselves the test either between or during sessions.

5. *MODEL OF USERS*

CLASSIFICATIONS OF USERS

This chapter defines and specifies the user model to be employed in initial tests. Some obvious extensions to the model are suggested, but they will not be used in initial testing.

Users must be characterized in ways that are felt to be important in predicting an optimum dialogue for their use of a particular service. Below is a classification along a single axis that is felt to be sufficiently comprehensive for this purpose, but it is certainly not exhaustive. Two other rather obvious dimensions are being developed but are not described in this report: they are functional (i.e., describing the user according to his job-related functional or operational behavior, such as clerical, professional, managerial, etc.) and demographic (i.e., describing the user according to his age, sex, experience, and so forth).

For test purposes in developing the theory, and for later application of the theory, users are preclassified off-line by administering standard psychological tests to them. Some of the tests must be conducted and evaluated by a trained, certified psychologist; hence application of the model for our initial experiment will be done by some qualified agency such as the USC Annenberg School of Communication.

PSYCHOLOGICAL CLASSIFICATION

The psychological classification defines and specifies the measurement of users' psychological attributes. These traits are of interest because of the importance of information processing behavior and ability in determining man-machine performance. Information processing ability is defined as the extent to which a user optimally encodes/decodes task information with as few errors as possible and within certain time constraints. Although many factors affect one's information processing ability, such as the amount of information to which one is exposed at any one moment, stress in the environment, role expectations, and so forth, information processing ability can be determined by the user's psychological traits, if one excludes environmental factors. The traits, and their measurement as defined below, are considered sufficient to predict behavior (users' performance) in these experiments. These tests were chosen primarily from the Western Psychological Services Catalogue [3] under the guidance of Dr. Gerhard Hanneman of the USC Annenberg School of Communications. Total test time prior to the experimental session is judged to be about one hour. Results are to be quantified as standard score centiles.

Skills and Ability

This test determines the extent of one's numerical ability, name matching, number digit retention (hence short-term memory) and similar abilities that are expected to be related to optimum user interaction with the man-machine language. The designated test is the *Hay Clerical Test Battery*.

Complexity Tolerance Level

This tests the extent to which an individual can process an increasing number of concurrent alternatives, for instance, examine a message while deciding which of several alternative commands would be most appropriate as the next input. This is expected to be a measure of one's ability to tolerate a certain level of uncertainty in the man-machine interaction--uncertainty which affects motivation, hence arousal, and similar characteristics impacting user's performance. The test chosen, which shows high correlation with these constructs, is ***Rokeach's Dogmatism Short Form.***

Spatial/Symbolic Reasoning

These constructs determine one's ability to solve problems, logically relate entities with similar attributes, reason at different levels of abstractions, and make analogies. It is an indicator of one's ability both to choose commands effectively and to interpret the man-machine dialogue with fidelity. The test designated is the Western Psychological Services ***WPS-EA 10, Symbolic Reasoning Test.***

Verbal Ability

This measures the understanding and range of one's vocabulary and the denotations and connotations connected with a symbol. One's verbal ability should affect efficiency in the man-machine dialogue. The designated test is the ***WPS EA 1 Verbal Comprehension Test*** of Western Psychological Services.

Manifest Anxiety

This measures the amount of residual day-to-day anxiety exhibited by an individual. Anxiety is related to frustrations, stress, and information overload thresholds. Not only is one's innate anxiety level of interest, but also the amount of anxiety induced by the man-machine dialogue interaction. An individual who evidences a high amount of anxiety should need more complete message feedback than would those showing low anxiety. The designated test is the ***Taylor Manifest Anxiety Scale,*** a derivative of the Minnesota Multiphasic Personality Inventory.

6. *MODEL OF SERVICES*

MODEL OF USER SENSITIVENESS

In its most useful form, the methodology being developed in this study will allow one to express criteria such as language forms in terms of predictors such as psychological user models and models of a planned service. This chapter describes a possible service model. Although only one service, military message processing, is being implemented for this study, the service model is nevertheless an integral component of the methodology. So, in one sense, the service model presented here is given for illustrative purposes. However, beyond its exemplification value, the User Monitor study expects later to develop an encoding scheme to quantify the application of the model to the message processing system. Furthermore, validity and reliability estimates of the quantification method will be determined.

A model of services is needed (along with a model of users) as a predictor in choosing language forms to achieve best user performance. The service is to be classified according to those characteristics that define an on-line, interactive service, but unusually and importantly, in terms of ***user sensitiveness***, that is, the way that the service affects the user and the accomplishment of his task. Like the psychological classification of the User Model, one should view this user sensitiveness classification as one of several reasonable dimensions that the Service Model might assume.

The service is preclassified off-line by an applications analyst. Each transaction type (i.e., command entry type, service operation, and response) is classified on a scale of 0 to 1 in each category below. Each category is an index or service descriptor. Below, under each index, the service property to be quantified is defined and examples illustrating the range of the scale are given. Values tending toward 1 indicate an increasing amount of positive user sensitiveness or system responsiveness; values approaching zero show a decrease in dynamics or accomplishment.

Directness Toward Goal

This property measures how closely the transaction type approaches the achievement of the user's general task, that is, an overall task such as sending a message as opposed to editing a line of text. This may be thought of as an index of user expectation. Sample transactions are described below which range from 0 progressively toward 1.

(i) Converse with the Tutor about an error condition. In this case the user is trying to correct a misunderstanding between himself and the service program. This operation has not ***directly*** come nearer to completing the general task.

(ii) Define a new dialogue element. In this instance the user has expanded his tools or facilities for future aid in completing the general task.

(iii) Prepare and edit a document. The user is directly approaching the general task more closely.

(iv) Send or receive a message. This operation completes an element of the task.

Complexity of Immediate Coal

This index quantifies the amount of abstract reasoning or judgment associated with a transaction type. Below are several examples ranging from 0 towards 1.

(i) Delete current line.

(ii) Replace all occurrences of string A by string B.

(iii) Send formal message requiring coordination.

Direction of Information Flow

This index measures the utility of the service to the user in terms of information transfer. Examples are given from 0 toward 1.

(i) Insert data.

(ii) Ask for a confirmation, such as intraline edit and type.

(iii) Interrogate, such as read mail.

Level of Memory Retention or Transaction Independency

This index measures the amount of latency of a transaction in relation to a sequence of transactions, that is, how much the user must recall about the context. Note that if arguments of the transaction type can be defaulted, this tends to lower the transaction type on the scale. Examples:

(i) Delete the line shown on the display terminal

(ii) Delete the next k lines

(iii) Purge file A.

Scale of Generality to Specialty

This index measures the focus of a transaction type, i.e., its range of applicability. Examples:

(i) Obtain approval to send a message

(ii) Set alerts on specified conditions

(iii) Edit a file.

Index of Ambiguity to Uniqueness

This measures the similarity of transactions in function (not in form). Where interpretation in context is required, the value tends toward 0. Those commands with singular, unique meaning tend toward 1. Note that if many parameters can be used the transaction tends to be lower on the scale than it would otherwise be.

6. Model of Services

Directness of Execution

This measures the straightforwardness of an operation. At the high end of the scale commands are executed directly, such as "send a message." At the low end, commands are indirect and are "assembled" for later use, such as in constructing a new language element.

Immediacy of Transaction

Timing measures the psychological expectancy of the anticipated response to a command. The examples range from 0 to 1

(i) Deferred (background), such as setting alerts for some future reminder.

(ii) Convenient (during session), such as retrieving from archive.

(iii) Conversational (several seconds), such as displaying a message from a list of messages.

(iv) Immediate (less than 1 second), such as intraline edit.

Transaction Usage

This index measures the expected relative frequency of use.
Examples range from 0 to 1.

(i) Low use, such as archive retrieval.

(ii) Medium use, such as message sending.

(iii) High use, such as message preparation.

Interaction Captaincy

Interaction captaincy measures the relative dominance between user and service. Service-driven, question-answer transactions are low on the scale, while operator-driven, demand-response transactions are high.

Component Quantity

This is a measure of complexity, but in terms of the multiplicity of parts of a transaction. Transactions allowing many arguments and providing many prompts tend toward 1.

7. LANGUAGE FORMS AND THEIR CONSTITUTION

FORMS OF DIALOGUE

Certainly one would not expect to alter a particular user's personal traits (type); similarly, a particular Service Model may be considered invariant during testing. (With the exception of hypothesis 1 stated in Chapter 3, user type remains constant.) The main variables discussed in this chapter, then, are the elements of dialogue by which the user and service communicate.

(As described in more detail in Chapter 4, several forms of man-machine language will be tested to validate some of the hypotheses stated earlier. A necessary preliminary step will be to pretest the languages through off-line protocol analysis to insure that no one favors the user's task, by design.)

Along with specifying (below) a set of language forms to be supported, a notation is introduced that will be useful in modeling the user's performance.

The input language forms supported are

1. Restricted English-like.

2. Function Key Operation.
   a. strictly function keys.
   b. function keys augmented with typed arguments.

3. Canonical.
   a. keyword functional notation.
   b. positional functional notation.

Notation: $I = \{ i1,...,i5 \}$.

The response forms supported are

1. English-like.

2. Menu for selection.

3. Forms for completion.

Notation: $O=\{ o1,o2,o3 \}$

REPRESENTATION AND TRANSFORMATION OF TRANSACTIONS

A ***dialogue element***, $t<i,o>$, is defined to be an element of $L = I \times O$. Obviously, for test purposes one need not support (nor is it feasible to support) the entire set, L. For example, menu selection and function key requests are generally considered complementary, whereas completing forms by function key is not. Thus, a subset (perhaps three) of combinations will be supported (for convenience, still called L), and the analysis will be limited to dialogue elements $t<i,o> \in L$.

The ***functions***, F, that a service can perform are initiated and completed by user/service transactions. One can denote a set of such functions by

$\{F<f> \mid 1 \le f \le n\}$, where each function corresponds to one or a series of service operations (requests).

A *transaction*, t<i,o,f>, has three phases: the I-phase or input portion of a dialogue element, the f-phase or function execution, and the O-phase or output. More specifically, during the I-phase the user may enter information and the CLP may respond directly to the user. This cycle may be repeated without the CLP having invoked the service. These responses are labeled *prompts*. After one or more such exchanges of information the I-phase is completed and the CLP invokes the service to begin the f-phase. The service is invoked to execute a *primitive function*. If the user-specified operation was a *compound function*, then the CLP issues the next primitive after the service responds to the CLP, and so forth until the operation is completed. Note that during the CLP/service interactions, the CLP does not interact further with the user. Upon completion of the f-phase the CLP then responds to the user (O-phase), which completes the life-cycle of a transaction.

View (I x O) x F as a matrix (See Fig. 3 and Chapter 11) whose row headings are the names of command/response pairs of I x O, i.e., the form of the dialogue, and whose column headings are the names of the primitive and compound functions. An element of the matrix (t<i,o,f>) represents an input form, a function, and an output form. Observe that one purpose of dialogue regulation (which multiple language forms allow) is to determine the best transaction for a given user/service situation. Thus the choices t<i,o,f>, $1 \le i,o \le m$ are provided for a given f. Since different transactions can invoke the same resulting function, one may define =, an equivalence relation on L x F that partitions the transaction types into equivalence classes [t<f>].

In summary, the service supports only one (consistent) internal form of each function, F<f>, corresponding to each equivalence class of transaction types. On the other hand, to individualize for users, different elements of each [t<f>] must be provided at the user's interface. The CLP performs both of these transformations. It performs transformations of I x O in a given equivalence class into the internal image F<f> of [t<f>] so that the service need support only one form while users may employ many forms. That is, α: [t] --> F is done by the CLP. Also, since users may change the dialogues, the CLP supports transformations among elements of each [t<f>], that is, β: [t] --> [t] is done by the CLP.

$$(t_{i,o,f}) = \begin{array}{cc} & \overbrace{\begin{array}{ccc} 1 & \cdots & \qquad\qquad\qquad n \end{array}}^{F} \\ \begin{array}{c} 1 \\ \vdots \\ m \end{array} & \left.\left[\begin{array}{c:c} \begin{array}{c}\text{language}\\ \text{primitives}\end{array} & \begin{array}{c}\text{language}\\ \text{compounds}\end{array} \end{array}\right]\right\} I \times O \end{array}$$

Figure 3. Transaction-type matrix

8. *PERFORMANCE INDICATORS AND DIALOGUE PROPERTIES*

MODELING OVERVIEW

With the goals, models, and transaction types in mind, let us identify additional modeling and measuring needs. Refer to Fig. 4, a schematic of the whole environment to be modeled. The outermost box represents the level at which one begins to formulate theory of dialogue. Given are the User Model and Service Model (both determined off-line) and the user's performance with respect to some dialogue form(s).

At the next (inner) level (in Fig. 4) we seek to 1) determine user's performance, 2) determine the relations between performance indicators and the properties of the dialogue, 3) test hypotheses concerning the significance of inefficient and ineffective dialogue, and 4) attempt to improve the dialogue on-line.

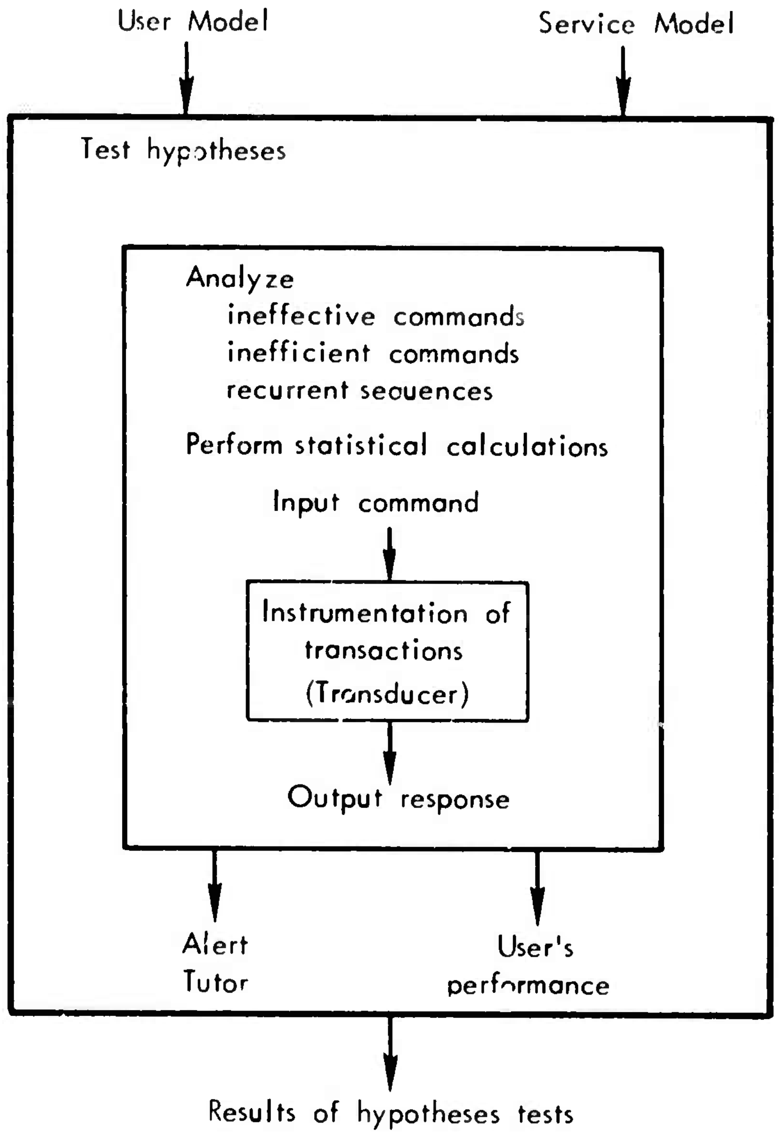


Figure 4. Dialogue theory modeling

8. Performance Indicators and Dialogue Properties

At the innermost level, one needs a concise way to think about and describe the instrumented transactions and their meaning with respect to level two. This, the actual man-machine interaction, may be thought of as a transaction transducer.

PERFORMANCE MEASURES AND MODELING

Performance Indicators

The observed performance indicators or measures are specified here. Chapter 11 describes the statistical analysis of these measures that comprise the user's performance model.

We are interested in measuring both the level of a user's performance and its rate of change. For example, in the first instance one might wish to compare two user groups using different fixed dialogue sets. In the second case, for example, we wish to meter the rate of improvement of a user as the language constructs he uses are altered to suit his preferences.

What then, are the metrics by which a user's performance can be gauged? Grossly, they are production, interaction rate, and errors. Following are measures of the nature of these metrics taken over each console session. Along with each sample measure, the circumstance under which it is acquired is given.

As a cross-check on the validity of these measures, a subjective quality measure is included.

Performance Indicators (production-related):

1. Volume of a message transmitted to hardcopy or consoles *other* than the user's. [The length is placed in a transcript buffer (See Appendix) by the Service Functional Module upon each such transaction.]

   The statistic calculated from this data is determined as follows. The range in volume of messages is calculated over all users over all sessions.

   R = M<H> - M<L>,

   where
   R = range,
   M<H> = length of longest message,
   M<L> = length of shortest message.

   Then the message length statistic, S1, is

   S1<i> = (M<i> - M<L>)/R,

   where M<i> = length of the <i>th message.

   Thus S1 is computed for each message for each user for each session.

2. Quantity of messages transmitted to hardcopy or consoles *other* than the user's. (This is determined by the presence or absence of a count in the transcript buffer.)

8. Performance Indicators and Dialogue Properties

The range of the number of messages sent by a user over a session is determined. Then the statistic S2 is determined similar to S1. This result is one S2 statistic per user per session.

3. Volume of messages received by the user. (This is likewise placed in the transcript buffer by the service routine.)

   This statistic S3 is calculated like S1.

4. Quantity of messages transmitted to the user's console. (See item 3 above.)

   The statistic S4 is calculated like S2.

Performance Indicators (work-rate related):

5. Relation of think time to cycle time. (Timing information is obtained either by the CLP or the Executive; it is placed in the transcript buffer for each transaction.) The transaction cycle time may be thought of as three ordered events: response to the previous input followed by user think time and entry of the next operation request.

   The statistic S5, think time divided by cycle time, is computed for each transaction that is sampled.

6. Relation of entry time to cycle time.

   The statistic S6 is calculated by dividing entry time by cycle time. It is computed for each sample.

Performance indicators (ineffectiveness-related):

7. Number of semantic errors detected in the service routine. (Upon each detection the service routine enters this information in the transcript buffer.)

   The statistic S7 is calculated like S2. The result is one statistic per user per session.

8. Number of right parses of user input by CLP. (This data is placed in the transcript buffer by the CLP.) The statistic S8 is calculated like S2.

9. Number of wrong parses of user input by CLP. (See item 8 above.)

Performance Indicator (quality-related):

10. The number of tasks completed correctly during the experiment.

    S10 is calculated like S2.

Notation: P={p<1>,...,p<10>} = performance indicator types S={s<1>,...,s<10>} = statistics of performance indicator types.

Note on Statistical Model of Performance

The statistical analysis which constitutes the user's performance model is detailed in Chapters 9, 11 and 12. Essentially, a set of statistics is determined from each of the statistics above, including mean, standard deviation, and variance of each performance measure, as well as validity and reliability checks.

INSTRUMENTATION OF TRANSACTIONS

Transaction Transducer

The fourth hypothesis implies that we wish to derive relationships between performance indicators and the formal properties of dialogue. Evidently, there is a group of data which represents the dialogue properties (related to each transaction) and which is associated with each performance indicator statistic, s<i,o,f>. It may help to visualize a transaction and its associated data by viewing transactions as performed by a transaction transducer. Intuitively, the transaction transducer waits for a command, accepts it, performs a function, and emits a response. L, defined earlier, is the language accepted by the transducer. A side effect of the transducer -- its central effect being the emission of a response to the user -- is to register the values of certain aspects of the dialogue and of the system. These aspects follow.

Formal Dialogue Properties

The data observed by the User Monitor during a sampled transaction are itemized below.

1. t<i,o,f>, code for command, function, and response,
2. Input command length,
3. Number of variables supplied,
4. Number of variables defaulted,
5. Number of prompts supplied,
6. Number of primitive function requests,
7. Length of canned response,
8. Length of data response to user,
9. Length of data response to other destinations,
10. Man-delay time.

Notation: D={d<1>,...,d<10>}= dialogue properties.

8. Performance Indicators and Dialogue Properties

Formal System Properties

The properties of the system that are of interest are listed below. In particular, time of day will be controlled in the initial experiments and the effect of load average will be eliminated. The values of these properties will be available in the transcript buffer for each transaction sampled.

1. Machine-delay time,
2. Date,
3. Time,
4. One-minute load average.

Notation: S={s<1>,...,s<4>} = system properties.

Hypothesis 3 stated that we wish to determine a small set of dialogue properties and performance indicators for future use. Similarly, we anticipate that load average will suffice in the future as the only necessary system variable to measure.

The Relation to Hypothesis 4

As stated above, these properties are recorded with each raw performance indicator sample; see Chapter 11. The preliminary statistical calculations follow closely those alluded to earlier for the performance measures. Except for the validity test described in Chapter 11, those tests are performed on the data collected for {d<1>,...,d<10>}.

Finally, in answer to hypothesis 4, we wish to study the joint distribution among variables in the performance indicator and dialogue property sets. The techniques used come under the heading of correlation, factor analysis, and regression are described in Chapter 12.

Relation to Hypothesis Set 6

Now that each model has been described somewhat, we return to the objectives implied in hypothesis set 6. Those hypotheses address issues of ideals in target languages by examining language extensibility and the most useful constructs. The performance, dialogue, and system data are the primary sources of information for study. From them one can readily answer the fundamental questions posed by these hypotheses.

9. *DIALOGUE REGULATION*

OVERVIEW AND MODEL

A primary goal of the User Monitor is dynamically to regulate dialogue forms (via the Tutor) to aid the user in improving performance by personalizing elements of the language. Three cases arise. The first is that in which the CLP is unable to recognize an input. The User Monitor attempts to identify the spurious command and suggest (via the Tutor) some remedial action. The second involves the isolation of those language constructs that do not prohibit useful work but that lead to poor performance. Again, alternate forms are recommended through the Tutor. The third is to recognize dialogue sequences which occur with significant regularity. Once identified, a composite form is suggested to the user.

This chapter discusses each of these three cases and, for each case, states an algorithm used to detect and remedy the "faulty" dialogue. In reading the following sections it might be helpful to visualize each case in the framework of the following model.

Let M = f (V<i>, N<j>),

where

M	= measure of the result of each alternative course of action
V<i>	= variables that specify alternative courses of action
N<j>	= states of nature, i.e., observations (or their statistics) of the man-machine interactions
f	= the functional relationship between the dependent variable M and the independent variables V<i> and N<j>.

According to this model, this chapter describes V<i> and N<j>; f and hence M come under the purview of the Tutor.

In the first case mentioned above, referred to as ineffective dialogue, N<j> denotes that a command was entered either correctly or incorrectly, and V<i> then specifies that either no action is taken by the User Monitor or that the command is analyzed according to the algorithm given and the results are passed along to the Tutor. In the second case mentioned above, referred to as Inefficient dialogue, the N<j> are defined to be 8 particular statistics, i.e., 4 tests based on confidence intervals of mean distributions of activity measures over all transacton types, and 4 within variance tests of individual transaction types, the V<i> are Tutor recommendations based on exceeding 2 standard deviation units in the confidence interval test or finding significant F-ratios of the variances. In the third case mentioned above, the N<j> are frequency counts of use of sequences of transactions and the V<i> are based on the resulting cumulative proportions.

Preceding page blank

INEFFECTIVE DIALOGUE ELEMENTS

Problem Statement and Overview of Solution Method

Ineffective (unrecognized) commands denote that class of errors which does indeed prohibit useful work, as opposed to other classes which simply lead to poor performance. These errors arise where the CLP has not been able to correctly recognize the attempted input command and the user has ultimately aborted the attempt by calling for help or the CLP has exhausted its guesses. At this juncture, the objective is to replace the problem-causing dialogue by a more useful form, thus eliminating the source of error.

Language improvement involves three steps. (1) The spurious command must be identified. (2) The cause of the error must be isolated to the extent of determining whether the problem is inherent in the input command or if the confusion arose from the previous output form. (3) The error source must then be eliminated by an appropriate change in dialogue form. The method outlined below for resolving unrecognized command errors is mainly based on analysis of two files which denote the frequency of use of transaction types, namely, the ***Transaction Relative Frequency Distribution Matrix*** (TRFDM) and the ***Recurrent Transaction Sequence Vectors*** (RTSVs); see Chapter 11. Knowing the user's current context, the last few transactions can be screened against the frequency distribution tables (RTSVs). The result provides an educated guess as to the spurious command's identity. Remedial strategies are similarly based on analyzing the knowledge and usage contained in these and similar tables.

Command Identification

The procedure follows for determining a statistical guess of the spurious command's identity; see Fig. 5. For each RTSV of entry length k, compare the previous k-1 successful transactions to each entry as follows. Compare the k-1st oldest transaction of the user's context with the first transaction of the sequence of an entry; the k-2nd to the second, and so forth through the last successful transaction to the k-1st transaction of the sequence in the table entry. If a match of the complete (k-1)-element sequence occurs, then note the sequence code, the last (kth) element of the sequence, and its cumulative proportion number (centile value). The kth element is potentially the spurious command we seek to identify. Continue with other entries in this manner until the table scan is exhausted.

After similarly processing each of the four tables (of sequence lengths 2, 3, 4, and 5), select several matches whose cumulative proportion numbers are highest, i.e., those least likely to occur due to random chance. If no matches were found, then the User Monitor has no suggestion for the Tutor as to the erroneous command's identity. If one or more matches are obtained, then the Tutor may converse with the user to see whether the command has been "guessed."

Once the command has been identified, the Tutor can determine the cause of error (i.e., this command or a previous response) by asking the user. At this point, the User Monitor can assist in dialogue form selection to circumvent further difficulty with the dialogue element.

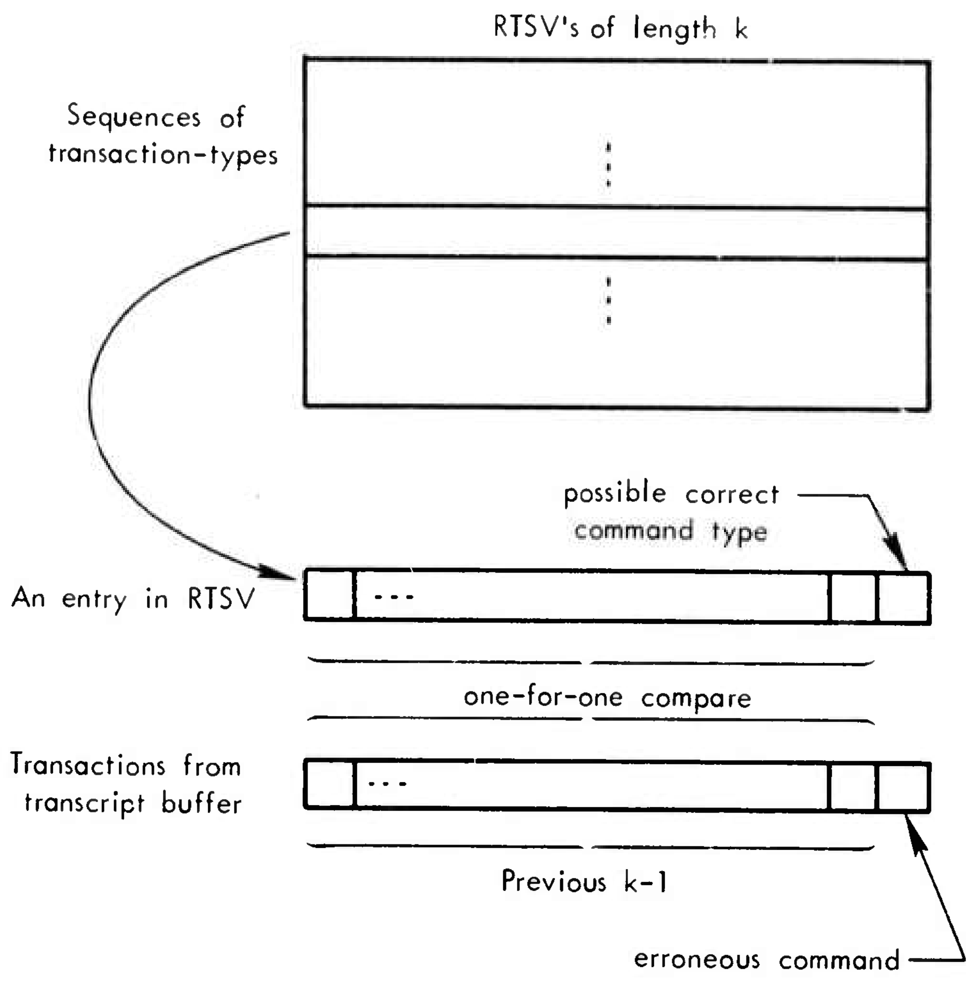

RTSV's of length k
Sequences of transaction-types
possible correct command type
An entry in RTSV
one-for-one compare
Transactions from transcript buffer
Previous k-1
erroneous command

Figure 5. Ineffective command identification

Dialogue Remedy

Given the identity of the command the user was trying to enter, the remedy depends upon whether he is having trouble with the identified command because of some attribute of the command itself or whether the trouble springs from the form of the previous response.

CASE 1: The input command is to be replaced. The questions to be asked and answered are stated below.

1. **Has he already created a composite command for the sequence? Determine by checking his extended TRFDM. If so, suggest that he use it. If not, proceed to step 2.**

2. Do other users have a composite command for this sequence? Determine by checking the aggregate TRFDM. If so, suggest their composite to this user.

3. Does his frequency of use of this command in the compound sequence at all warrant a composite, i.e., does it pass a less rigid confidence interval test than normally required for suggestion, say 90%? If so, suggest it.

If usage does not justify a transaction defining a compound function, then the next approach is to attempt to change the transaction form, i.e., suggest another transaction type for tho same function. The selection procedure follows:

1. Calculate the max row sum of t<i,o,f>, (i,o)=1,...,m, f=1,...,n, f≠(i,o) of his TRFDM. That is, determine his most frequently used form different from the current form. (If all sums are identically zero, go to step 2.) Suggest the transaction form {t<i,o,f>}, i.e., the same function he was trying to invoke but by the dialogue form he most frequently uses, other than the one in error.

2. If all sums above are zero, then suggest the form most frequently used by the collective user population for this function. Using the aggregate TRFDM, calculate the max {t<i,o,f>}, (i,o)=1,...,m, (i,o)≠f. That is, determine the most frequently used element of [t<i,o>]. Suggest this {t<i,o,f>} as a replacement form.

CASE 2: The previous response form is to be replaced.

1. Perform sequence analysis as in Case 1.

2. Conduct dialogue element replacement as in Case 1 but with the following restriction: in steps 1 and 2 of Case 1, Part 2, examine only rows, and the jth column, respectively, for which the input form of {t<i,o,j>} matches. Since the input form of the last successful transaction, {t<i,oj>}, caused no difficulty, we wish to choose a replacement from the restricted set whose input form is the same as that of the transaction being replaced.

INEFFICIENT DIALOGUE ELEMENTS

Inefficient commands are those which do not prohibit useful work, but do lead to poor performance. Such commands are ferreted out during regular post-session analysis. Findings are reported to the Tutor via the Potential Dialogue Improvement (PDI) file (see Chapter 11) when the user logs on again.

Analysis is similar to that involved in determining the user performance model, where calculations on s<i,o,f> range over all commands. However, to detect below-par performance of individual commands, calculations are done with respect to each language form of I x O. In general, according to the performance criteria named below, we examine the mean and variance of the performance of each command type in relation to other command types to find those forms leading to poor performance. A high variance suggests that the dialogue element is being applied for multiple semantic purposes. The Tutor might suggest that the user build a second command, ixoxf' where f'=f, functionally, but with a different name, and the I x O is of the same syntax as before.

9. Dialogue Regulation

As a result of mean and variance tests the PDI file can contain poor performance dialogue elements in three cases. For a given entry, if the mean is significant but the variance is not, then the Tutor will suggest a one-for-one replacement. If the variance is significant, regardless of whether or not the mean is significant, the Tutor will suggest splitting the command as indicated above.

The following four efficiency indices are calculated for each transaction sampled during the session.

1. Interaction rate (machine time divided by man time plus machine time).

2. Parameter utilization (number of arguments supplied plus one, divided by number of arguments supplied and number defaulted plus one).

3. Entry accuracy (inverse of number of correct parses plus one).

4. Entry inaccuracy (inverse of number of incorrect parses plus one).

Poor Performance Algorithm

1. Calculate efficiency indices from the TRS file; see Chapter 11.

2. Calculate the mean and variance of each efficiency index over all t<i,o,f>.

3. Replace each efficiency index statistic by its standard score.

4. For each t<i,o,f> compute the mean over all standard scores of all indices.

5. Compare these means to the interval -1.96 to +1.96 for 95 percent and to -2.58 to +2.58 for 99 percent.

6. Enter high and low means in the PDI file.

7. Compute total sum of squares for variance.

8. Compute "between" sum of squares.

9. Compute "within" sum of squares.

10. Determine degrees of freedom.

11. Calculate mean squares and the F-ratio.

12. Use F-table to determine significant variance.

13. Enter significant variances in PDI file.

RECURRENT DIALOGUE SEQUENCES

One phenomenon of interest in increasing performance is to detect when the user does things habitually and then take the initiative in reducing the amount of protocol necessary for him to accomplish that particular task. That is, if he repeatedly types the same several commands in sequence, we would like to replace the sequence by a single composite operation. For example, the user might consistently ask for a retype of the line he has just edited, or he may frequently read his mail directly after he has logged on in the morning.

The User Monitor requests that the CLP provide the transaction data (i.e., contents of the transcript buffer) on the next n (approximately 10) transactions.

That is, we wish for the Tutor and user to define a new transaction type to replace a sequence. Notice that such a definition operation extends the columns (functions) of the TRFDM by adding a compound-function column.

The following procedure is used to analyze the transaction sequences:

1. Sample randomly, taking approximately 10 successive transactions as a sample.

2. Form separate frequency distribution tables for sequences of lengths 2, 3, 4, and 5.

3. Treat as binomial data. Compute mean and standard deviation.

4. Convert to standard scores.

5. Determine cumulative proportions based on frequency of occurrence and enter them in the PDI file.

See Chapter 11 for a more complete specification of format of the frequency tables and the algorithm used for composing and maintaining them.

10. *PROGRAM ORGANIZATION AND CONTROL*

The User Monitor's control flow is outlined below; its task is essentially as follows. While the user is in session, data on transactions are gathered for later analysis; those commands that cannot be recognized by the CLP are also analyzed on-line. When a session is completed, the collected data are analyzed to discover recurrent transaction sequences and dialogue elements whose use results in poor performance. After the on-line experiment is completed, an analysis program is run to confirm or reject the hypotheses.

OUTLINE OF CONTROL FLOW

1. When user logs on:
   a. Do housekeeping, e.g., load user profile.
   b. Tell Tutor about possible new commands arising from both poor performance and repeated dialogue sequences.
   c. Start pseudo-random-sampling clock.
2. While user is in session, in real time:
   a. Take random samples and write Transaction Raw Sample (TRS) file (see Chapter 11).
   b. Perform ineffective command analysis.
3. When user logs off:
   a. Write user profile.
   b. Schedule profile update and analysis of inefficient dialogue and recurrent sequences.
4. Post-session profile update and analysis:
   a. Merge new session data from TRS into Transaction Relative Frequency Distribution Matrix (TRFDM) and delete oldest session data from TRFDM; also update Aggregate TRFDM (ATRFDM).
   b. Likewise update Recurrent Transaction Sequence Vectors(s) (RTSVs), perform RTS analysis, do same for Aggregate RTSV (ARTSV), construct Potential Dialogue Improvement (PDI) list for Tutor.
   c. Perform user performance analysis and write User Performance Statistics (UPS) file.
   d. Perform poor-performance analysis and append results to PDI for Tutor.
   e. Write session summary file for analyst.

10. Program Organization and Control

5. Hypotheses Tester:

   a. Run hypotheses tester at completion of experiment.

MAJOR MODULES

The more prominent User Monitor program modules are the following:

1. Real-time initialization
2. Pseudo-random sampler
3. Ineffective dialogue analysis
4. Real-time post processor
5. User profile update
6. RTS analyzer
7. Poor-performance analyzer
8. Performance indicator calculator
9. Analyst's summary of session module
10. Hypotheses tester
    - driver
    - analyst's output
    - basic statistics calculator
    - variance analyzer
    - canonical correlation analyzer
    - time series analyzer
    - regression fitter
    - discriminant analyzer

11. *DATA BASES*

This chapter defines the concomitant data bases essential to the operation of the User Monitor, et al., states their principal use, specifies their general organization, and indicates the procedures for their construction and maintenance.

USER PROFILE DATA

The User Profile contains information about the user's knowledge and recent use of a service. As a concise record of the user's behavior and experience with the message service, it serves as the message service environment's memory for discourse with the user on a personal level. The files comprising the profile are named below; they (and other files) are subsequently described in more detail.

1. Transaction Relative Frequency Distribution Matrix (TRFDM),
2. Recurrent Transaction Sequence Vectors (RTSVs),
3. Potential Dialogue Improvement (PDI) File,
4. Transaction Training Statistics (TTS) File,
5. User performance Statistics (UPS) File.

TRANSACTION RAW SAMPLE (TRS) FILE

Use

The TRS file contains data samples of user/service transactions. It is the source of raw data used to compile statistical files that are subsequently analyzed in search of improvements in dialogue and dialogue forms. This in turn should result in improved user performance.

Format

Each TRS contains some number of sample records taken over one session; each record contains the observations of ten successive transactions. Transaction data for each block of a record contains the following information:

1. The formal dialogue properties.
2. The formal system properties (omitting property 2).

Header information of the TRS file consists of:

3. User identification code.
4. Date or TRS session sequence number.
5. Session log-on and log-off times.
6. Indicator as to whether or not statistics computed from this TRS have been incorporated into the User Profile.
7. Indicator as to whether or not statistics computed from this TRS have been expunged from the User Profile.

Construction Procedure

Measurements are taken from the transcript buffer at times determined by pseudo-random sampling. The TRS is written as samples are taken. Generally, the number of samples taken over a session is a Bayesian function of sampling cost (to consume approximately 1 percent of the system resources absorbed by a particular user) and the penalty of making incorrect inferences based on too small a sample size, but in the case of the first experiment (see Chapter 4) all transactions may be sampled.

TRANSACTION RELATIVE FREQUENCY DISTRIBUTION MATRIX (TRFDM)

Use

The TRFDM specifies the relative frequency of use of each transaction type. It is used as a source of familiar dialogue forms in replacing both inefficient and ineffective dialogue elements, and it is also used by the Tutor for training purposes. The TRFDM represents current use of a service in that counts reflect only the past j sessions.

Format

The TRFDM is an m x (n + k) array of usage tallies, where m is the dimension of language forms, n is the number of primitive functions, and k is the number of compound functions.

Construction Procedure

The matrix entries are formed from a simple tally of the transaction samples in the TRS files. The statistics are normalized with respect to sample size so that they may be interpreted in a consistent way by the Tutor.

AGGREGATE TRFDM

Use

The ATRFDM may be used as a source (secondary to the TRFDM) of familiar dialogue forms for replacing ineffective and inefficient dialogue elements. ("Aggregate" denotes that the counts are summed over all subjects that are members of the same user and service classes.)

Format

The format is very similar to the TRFDM.

Construction Procedure

The ATRFDM is maintained from the individual TRFDMs, and is normalized with respect to sample size.

RECURRENT TRANSACTION SEQUENCE VECTORS (RTSV)

Use

The RTSVs show the relative frequency of use of sequences of transaction types. They are used to select prominent recurrent sequences as suggested replacements by new transaction types that execute compound functions.

Format

1. There are four such vectors representing, respectively, sequences of lengths 2-5. Each vector is fixed length (jx1), which implies that as new sequences are added, ones of lesser significance are removed. Like the TRFDM, the RTSVs reflect recent interaction, i.e., the last n sessions. The vector header identifies the sequence length and contains the sample mean. Entries are partially ordered on decreasing cumulative proportion values.

2. Each entry is formatted as 2k+6 cells, where k is the length of the sequence.
   a. IxO (dialogue. numbers.
   b. F (service function) numbers
      (the a and b 3-tuples are given for each
      transaction type of the sequence of length k).
   c. Raw relative frequency count.
   d. Standard score frequency count over all RTSVs,
   e. Standard deviation of this sequence based on
      standard score distribution.
   f. Cumulative proportion.
   g. Number of times compound function
      has been suggested to user.
   h. Indicator as to whether or not the sequence has
      been adopted in composite form by the user.

Construction Procedure

The procedure for constructing the RTSVs is as follows. Note that the data source is the last n sessions of the TRS.

1. Compute mean and standard deviation of sequence frequencies for the sequence names.

2. Transform the frequency scores to standard scores.

3. Compute the standard deviation for each sequence name, based on the transformed distribution.

4. Compute cumulative proportions.

5. Order each RTSV set on decreasing statistical significance. (Note: the RTS analysis procedure examines the RTSVs and makes entries in the PDI file.)

AGGREGATE RTSVs (ARTSVs)

Use

The ARTSVs provide a secondary source of recurrent transaction sequences. They are used similarly to RTSVs. The word "aggregate" denotes that these vectors represent distributions over all users in the same user/service class.

Format

This format is similar to that of the RTSV.

a. Sequence of codes (I x O x f).

b. Range and mean of standard scores.

c. Composite standard deviation and cumulative proportion.

d. Number of users who know the compound function represented.

e. Number of users employing the compound.

Construction Procedure

Since the frequencies are standardized, the ARTSVs are calculated similarly to the RTSVs.

POTENTIAL DIALOGUE IMPROVEMENT (PDI) FILE

Use

The PDI file denotes significant recurrent sequences and also poor performance dialogue elements. It is maintained by the User Monitor. The Tutor uses the PDI in training the user at the beginning or end of a session.

Contents

1. The first record contains relative addresses of RTSV entries that surpass the 99% and 95% confidence limits, i.e., candidates for compound operations. This record also contains similar relative addresses of ARTSVs. Both sets are ordered on decreasing statistical significance.

2. The second record lists the t<i,o,f> resulting in poor performance. (Note that as these are presented to the user, the Tutor employs the User Monitor to determine replacement strategies as defined in the Dialogue Regulation chapter.) This list is ordered on increasing performance.

   Each element of the list contains indicators that tell whether the element was included because of significant difference in mean or variance. The Tutor uses this information to determine whether to suggest a one-for-one or a two-for-one replacement of the dialogue element. See Chapter 9.

11. Data Bases

TRANSACTION TRAINING STATISTICS (TTS) FILE

Use

The TTS file, a component of the User Profile, is maintained by the Tutor and used as reference in teaching the user.

Format and Contents

The dimensions correspond to the TRFDM. An entry contains the following:

1. Each element contains an indicator of whether the user has been taught the dialogue element by the Tutor.

2. Each element denotes the amount of training.

3. Each element specifies the kind of training.

USER PERFORMANCE STATISTICS (UPS) FILE

Use

The UPS file contains statistics calculated from the performance indicators. It is used primarily as a data source for testing hypotheses, as well as by the Tutor. It represents a statistical model of a user's performance.

Contents and Construction

The following statistical calculations suggest the contents, construction, and some uses of this file. From each session a set of statistics is obtained, s<i,j>, 1≤I≤9, where j = number of events monitored for each i.

1. Make up frequency tables (over the j's) to determine the distributions of each s<i>.

2. Calculate the means of s<i>,i=1, ..., 9 to get an idea of typical performance.

3. Determine the standard deviation and variance of s<i>, i=1,...,9.

4. Code the above information into standards for comparisons among groups.

5. Determine relationships of one measure of performance, s<i>, to another measure of performance, s<k>.

6. Determine reliability of measuring techniques by correlating two sets (sessions) of measurements over all s<i>.

7. Determine the validity of measurements by (item 5) and by subjective comparison. See Chapter 4.

8. Relate one user's data with another's to determine where an Indicator can be applied to a third user. This Applies to users of the same group.

9. Make off-line inferences about the population (i.e., generalize) on the basis of the analysis of a few samples.

COMPOUND DEFINITIONS FILE (CDF)

Purpose

The CDF displays macro commands that have been defined by the user. It is constructed by the CLP as macros are defined, and it is used by the User Monitor for various analyses.

Format

The header contains a 3-tuple (i,o,f) which is the internal name of the macro. The body is an ordered list of the 3-tuples (i,o,f) which make up its definition.

12. *ANALYSES*

ANALYSIS TECHNIQUES USED

Some classical techniques of statistical analysis [4,5] are repeatedly used in testing the hypotheses stated in Chapter 3. The kinds of estimates obtained with each method is briefly reviewed below. A more complete description can be found in most standard works on statistics.

Analysis of Variance

Analysis of variance is essentially a method to separate and estimate a number of sources of variation to which observations are subject. The particulars of the procedure, with regard to given hypotheses, depends on the number and the nature of the independent sources of variation. The analysis of variance, for our purposes, can be viewed as two similar problems, distinct primarily in their interpretation. The first problem, called estimation of variances, is to estimate causes of variance of a population from samples taken from that population. Where significant variances are determined, the second problem, called comparison of means, makes specific comparisons between observations or their statistics.

The bulk of our analysis will involve the analysis of variance.

Discriminant Analysis

Discriminant analysis tests whether or not significant differences exist among the average scores of some test variable, say language forms. If so, linear combinations of the predicator variables (say, performance indicators) are derived to enable the analyst to represent language forms by maximizing among-group variation relative to within-group variation.

Time Series Analysis

A time series is an ordered set of observations, ordered on time of taking the samples. Analysis, in our case, merely involves something like a least-squares fit for graphing the results. This technique is used sparingly and is accompanied by some aspects of analysis of variance.

Linear Regression Analysis

Linear regression allows one to determine the effect, and its amount, of one quantity upon another. Essentially, we estimate empirical relations between variables, and our principal objective is to find functional relationships, where they exist.

Regression specifically differs from correlation (below) in that with regression one criterion (dependent) variable is to be expressed in terms of a number of predicator (independent) variables. Again, the purpose is to derive an equation of regression that permits us to estimate the value of one variable given the values of others.

12. Analyses

Correlation Analysis

Correlation is a multivariate analysis technique in which we measure the relationship between sets of criterion variables and sets of precicator variables. Unlike regression, where the value of one variable is estimated from the values of others, correlation determines which variables in each of the two sets contribute most to the association between the sets.

This is a weaker form of analysis than regression or discriminant analysis; hence, its use will be limited in our studies.

Factor Analysis

Factor analysis is really a collection of statistical techniques aimed at data reduction. One does not normally partition into criterion and predicator sets as in correlation analysis. In fact, though, for our purposes, separate factor analyses will be performed on partitioned sets in order to establish the strength of overall relationships among variables within the sets. The purpose is to reduce a large number of variables to a small set of linear combinations of those variables.

Factor analysis is not used very heavily in our analysis.

TEST OF HYPOTHESES

The remainder of this chapter identifies the statistical techniques to be used in testing each hypothesis. In most cases, a figure illustrating the necessary data organization accompanies the description.

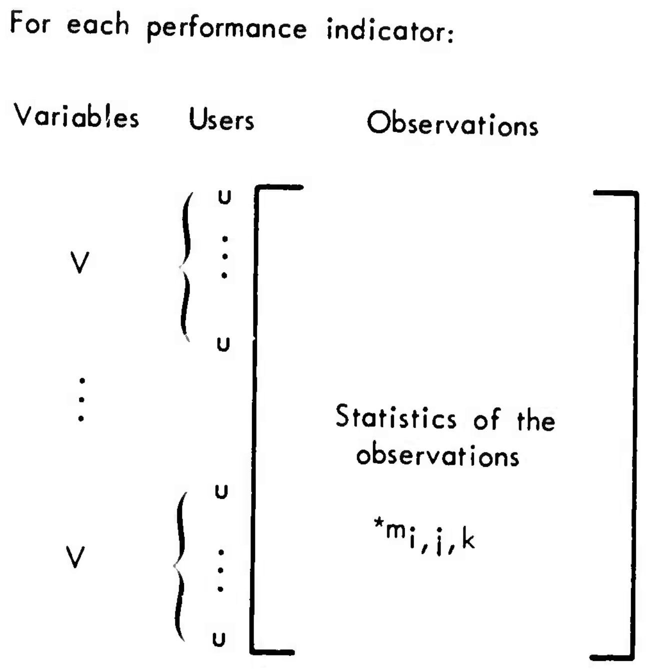
Analysis 1: User Model

Hypothesis 1 is to be examined using the techniques of analysis of variance. The data organization is quite similar to that in hypothesis 2, hence refer to Fig. 6. In Fig. 6, replace "V" by "user type."

Analysis 2: Overall Performance

Hypothesis 2 is to be analyzed using one-way analysis of variance. Both estimates of variances and comparison of mean values will be made. Each of these three analyses involved hierarchic classification of data with three sources of variation, viz., within user, between users within groups, and between groups.

Figure 6 Indicates the data organization for which the analysis will be made for each performance indicator. The performance indicators are intentionally not combined in any way, so that the results can be weighted appropriately in any future application of them.


For each performance indicator:
Variables
Users
Observations
V
U
U
V
U
U
Statistics of the observations
*$m_{i,j,k}$

*i = observation
j = user
k = language form

Figure 6. Data organization for overall performance.

Analysis 3: Maximum Discriminants

Linear discriminant analysis is applied to language forms. The data organization for the analysis is shown in Figure 7.

$$\begin{array}{cc} & \text{Performance indicators} \\ \begin{matrix}\text{Language} \\ \text{forms}\end{matrix} & \begin{bmatrix} \text{Means of} \\ \text{performance} \\ \text{indicators} \end{bmatrix} \end{array}$$

Figure 7. Data organization for maximum discriminants

Analysis 4: Formal Dialogue Properties

Hypothesis 4 can be tested either by factor analysis followed by regression or by correlation. Figure 8 illustrates the data set up for canonical correlation analysis. Analyses will be performed for each language form group and for all users combined.

$$\begin{array}{cccc} & \text{Users} & \text{Performance indicators} & \text{Dialogue properties} \\ \begin{matrix}\text{Language} \\ \text{forms}\end{matrix} & \left\{\begin{matrix} u_1 \\ \vdots \\ \\ u_n \end{matrix}\right. & \begin{bmatrix} \\ \text{Means} \\ \\ \\ \end{bmatrix} & \begin{bmatrix} \\ \text{Means} \\ \\ \\ \end{bmatrix} \end{array}$$

Figure 8. Possible test for dialogue properties

Analysis 5: Relationship of Language Forms to Training

First a time-series analysis is performed by fitting a trend line. Then either regression or analysis of variance is used to measure differences. Data must be transformed because of nonhomogeneous variance due to differences in experience as the experiment progresses.

12. Analyses

The following are graphed for hypothesis 5. Plot each performance indicator separately (see Fig. 9) for each group of the variable being tested. Data points are means (separate plots are made where they are variances) of the group, over a session. Time divisions are console sessions.

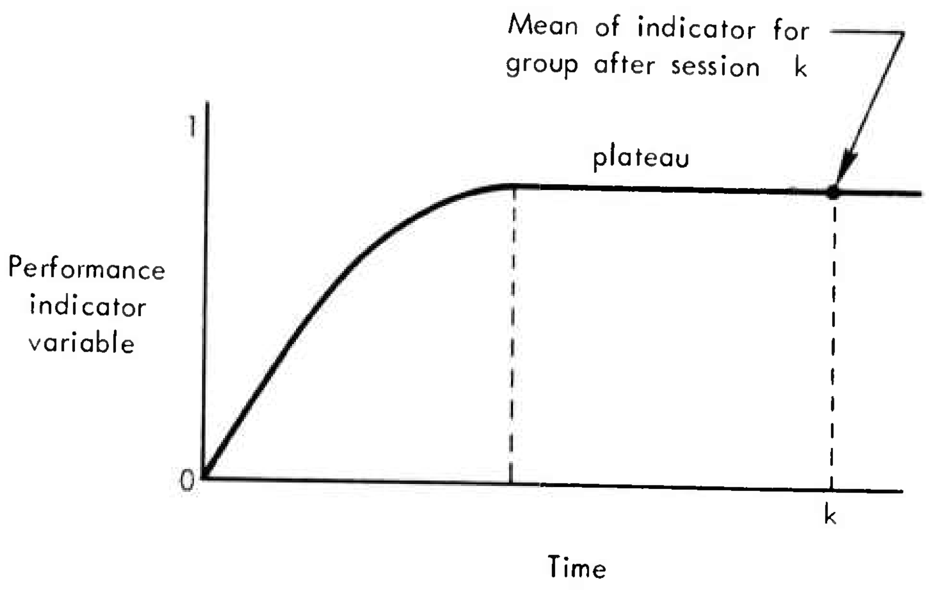


Figure 9. Time series graph of language forms

Analysis Set 6: Language Extensions

Hypothesis 6A is checked by analysis of variance: see Fig. 10 for data organization. In a similar fashion, hypothesis 6B is tested by analysis of variance: see Fig. 11. Hypothesis 6C is checked like 6A with the language form variable replaced by experience level.

Hypothesis 6D is tested by graphing rank ordered items (or their inverse, which gives cumulative counts) where items are frequency of use of commands. Then a CHI square (or analysis of variance) test is done on the number of transaction types that account for, say, 95% of the use. A graph and a test is constructed for each language form.

Hypothesis 6E is tested by first performing a factor analysis followed by regression. Figure 12 shows the data organization for factor analysis.

Hypotheses 6F and 6G are tested by graphing a least-squares curve fit of frequency of use versus length of compound element (6I) or length of time it has existed (6G). Frequencies are normalized on the basis of time that the compounds have existed. The data will be plotted before and after transformation.

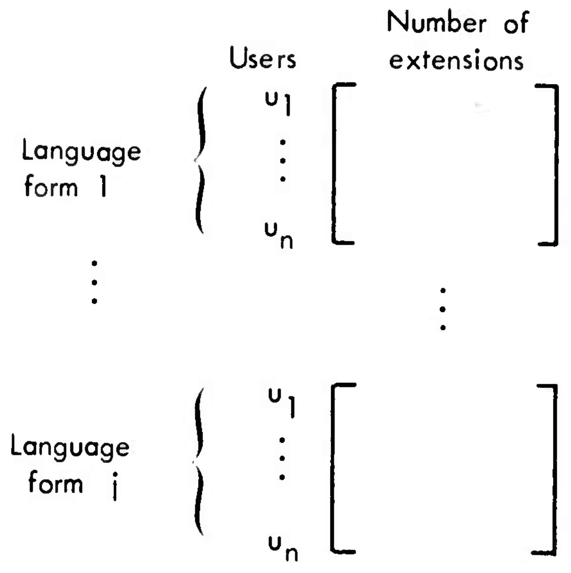


Figure 10. Data organization for hypothesis 6A

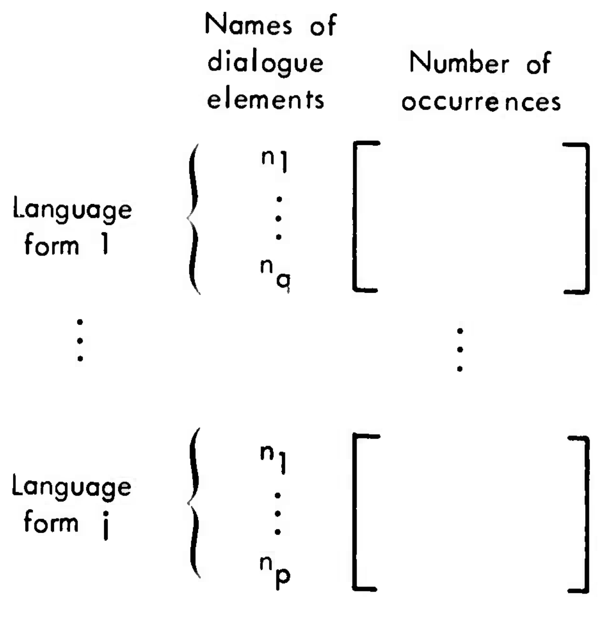


Figure 11. Data organization for hypothesis 6B

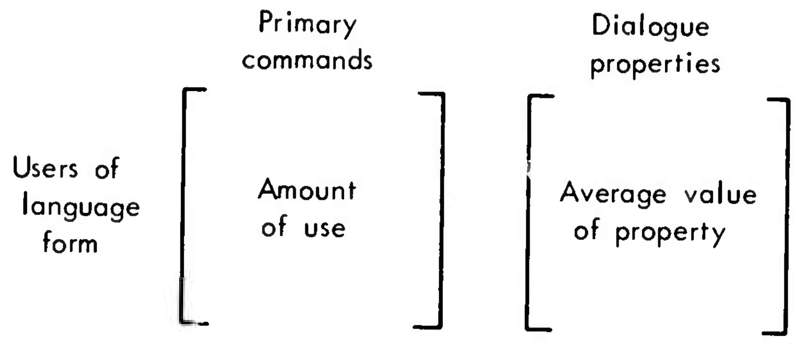


Figure 12. Data organization for 6E factor analysis

Analysis Set 7: Effectiveness, Efficiency, and Recurrency

These hypotheses are tested using analysis of variance. Data organization for 7A is shown in Fig. 13. An analysis is performed for each language form and for each statistic indicated by the column headings. Likewise, the data organization for hypotheses 7B and 7C are illustrated in Fig. 14.

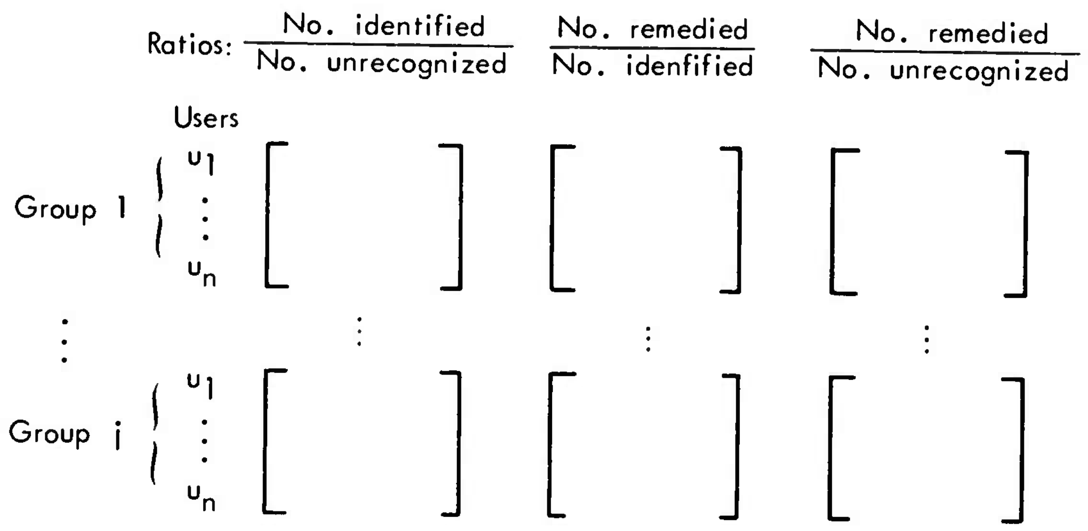

Ratios:
No. identified / No. unrecognized
No. remedied / No. idenfified
No. remedied / No. unrecognized
Users
Group 1
u1
un
Group i
u1
un

Figure 13. Data organization for hypothesis 7A

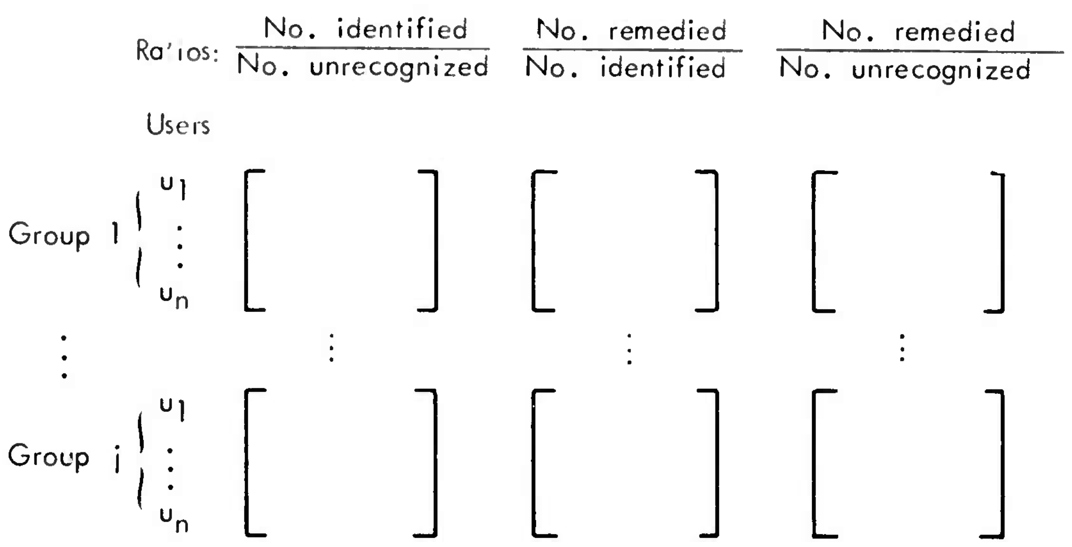

Ratios:
No. identified / No. unrecognized
No. remedied / No. identified
No. remedied / No. unrecognized
Users
Group 1
Group j
u1
un

Figure 14. Data organization for hypotheses 7B and 7C

APPENDIX: TAXONOMY

The following words and phrases are idiomatic terms used in the description of the User Monitor. Some of them carry over their meaning to other components of the Information Automation project.

Command Language Processor (CLP) - a program element of the whole message processing system that acts as a transparent (to the user) intermediary between the user and the message service program.

Composite Command- See Compound Function.

Compound Function- an ordered sequence of service operations (see function and primitive function) invoked by a single macro input request. The macro or compound has been defined by the user to "extend" the dialogue element set.

Compound Functions Definition (CDF) File - a data base of definitions of language extensions in terms of macro commands.

Criterion Variable- a dependent variable, denoting class membership, expressed as a function of predictor or independent variables. See Predictor Variable.

Dialogue Element- a single man-machine communication form, characterized by its two components, viz., a form of input, command, or request such as functional notation, and a form of output or response such as a menu for selecting among alternative inputs.

Dialogue Properties- See Formal Dialogue Properties.

Executive - the program or resource management program of the message processing system that pertains only to message processing. That is, it runs as a job under the operating system.

Formal Dialogue Properties - attributes of the man-machine communication that characterize dialogue and its periphery in terms of its format and complexity, such as the number of parameters supplied on input and the user's think time before command entry.

Function - an operation or procedure conducted by the message processing service, invoked by a command from the user.

Ineffective Dialogue Element - a dialogue element whose input component is repeatedly entered incorrectly by the user to the point of exhausting the possible correct interpretations by the CLP. Such an element is so denoted if, after several tries, the user abandons the attempt at correct recognition and asks the Tutor for help.

Inefficient Dialogue Element - an element of dialogue that does not prohibit useful work, but does lead to poor performance according to specific criteria defined in this report.

Appendix A: Taxonomy

Information Automation project	an ARPA-sponsored study at ISI initiated to define the methods needed to fruitfully automate large sectors of military communications. As part of the study, an experimental message processing system is to be provided on the ARPANET.
Language Form	a form of man-machine communication with a single mode or form of input and a single mode or form of output. For a particular service, a language form is composed of a collection of uniform dialogue elements to invoke each service function. See Dialogue Element and Service Functions.
Message Processing Service	an experimental, man-machine service being developed for the Information Automation project (q.v.). In particular, reference within this report is to the program which actually implements the message processing functions.
Message Processing System	includes the whole of the IA message processing service and other program modules, as well as the operating system and equipment out to and including the man-machine terminals.
Performance Indicator	a metric by which user's performance is gauged in on-line interactions. Such indicators measure rates of production, interaction, and errors.
Poor Performance Dialogue	See Inefficient Dialogue Element
Potential Dialogue Improvement File (PDI)	a data base listing the most significant recurrent transaction sequences and inefficient dialogue elements.
Predictor Variable	an independent variable. A linear combination of independent variables is evaluated to a numerical criterion to predict membership in a class. See Criterion Variable.
Primary Language Element	a member of that class of dialogue elements (both primitive and compound) that accounts for 95% of service usage.
Primitive Function	an operation (see Function) that is invoked by a single user's input request in the language initially supplied to the user. See Compound Function.
Prompt	an intermediate response from the Command Language Processor to the user, provided to assist the user in entering a proper command.
Recurrent Dialogue Sequences	ordered sequences of dialogue elements that are repeated by the user or service to the extent that their occurrence is statistically significant.
Recurrent Transaction Sequence Vector (RTSV)	a data base specifying the relative frequency of use of sequences of transaction types.

Appendix A: Taxonomy

Service Function- See Function.

Service Model - a conceptual framework to express the characteristics of the message processing service--in our case, characteristics are expressed in terms of user sensitiveness--and to quantify those characteristics in order to relate them to other phenomena, such as to relate them to language forms via users' performance.

Service Program- See Message Processing Service.

System Properties- those environmental attributes of the man-machine communication system, such as average computer load and time of day, that are to be either controlled or normalized in the project experiment.

Transaction - an instance of use of a Transaction Type (q.v.).

Transaction Cycle- an ordered event sequence comprising a single man-machine transaction. The order is response from previous request, man's think time, and command entry time.

Transaction Raw Sample File (TRS) - a data base containing raw data, from sampled transcript buffers, on performance indicators, dialogue properties, and system properties.

Transaction Relative Frequency Distribution Matrix (TRFDM) - a data base specifying the relative frequency of use of each transaction type.

Transaction Training Statistics File (TTS) - a data base denoting the kind and amount of training that the user has received with respect to each transaction type.

Transaction Transducer- a conceptual, automata-theoretic model of man-machine transactions.

Transaction Type- a single kind of an element of man-machine communication. It is described by a 3-tuple which consists of an input request, an output response, and a service function. See Dialogue Element and Function.

Transcript Buffer- a circular storage buffer maintained by the Command Language Processor and other program modules, which contains values pertinent to the most recent man-machine transactions, such as the kind of transaction, how long it took to enter, and so forth.

Transducer - See Transaction Transducer.

Tutor - a program element of the whole message processing system, that provides introductory training and on-line assistance to the user in helping resolve errors and ambiguities.

User Model - a conceptual framework to express the characteristics of a user--in our case, characteristics are psychological traits--and to

	quantify those characteristics in order to relate them to other phenomena, such as to relate them to language forms via users' performance.
User Monitor	- a program element of the entire message processing system, which measures user/service interactions and later analyzes the measurements to test hypotheses basic to the development of the Information Automation methodology. It also contains an interactive component to help resolve users' errors and ambiguities.
User Performance Statistics File (UPS)	- a data base summarizing the statistics of performance indicators.
User Profile	- a collection of data sets, whose contents are unique to each user, that describes the user's knowledge, performance, training, and other dialogue preference idiosyncrasies. Chapter 11 describes each constituent data set of the User Profile.

REFERENCES

1. Oestreicher, D. R., J. F. Heafner, and J. G. Rothenberg, "CONNECT: A User-Oriented Communications Service," To be presented at ACM Annual Conference, November 1974.

2. Ellis, T. ., L. Gallenson, J. F. Heafner and J. T. Melvin, ***A Plan for Consolidation and Automation of Military Telecommunications on Oahu***, May 1973, USC/Information Sciences Institute, ISI/RR-73-12.

3. ***Western Psychological Services Catalog***, Western Psychological Services, 12031 Wilshire Boulevard Los Angeles, California 90025

4. Davies, O. L., ***Statistical Methods in Research and Production***, Hafner Publishing Company, New York, 1958.

5. Green, P. E., and D. S. Tull, ***Research for Marketing Decisions***, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

BIBLIOGRAPHY

1. Abbott, R. J., "A Command Language Processor for Flexible Interface Design," ISI/RR-74-24.

2. Mandell, R. L., "An Executive Dosign to Support Military Message Processing Under TENEX," ISI/RR-74-25.

3. Rothenberg, J. G., "An Editor to Support Military Message Processing Personnel," ISI/RR-74-27.

4. Rothenberg, J. G., "An Intelligent Tutor: On-line Documentation and Help for a Military Message Service," ISI/RR-74-26.

5. Tugender, R., and D. R. Oestreicher, "Basic Functional Capabilities for a Military Message Processing Service," ISI/RR-74-23.